

# SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment

Grace A. Lewis  
Edwin J. Morris  
Dennis B. Smith  
Soumya Simanta

June 2008

**TECHNICAL NOTE**  
CMU/SEI-2008-TN-008

**Integration of Software-Intensive Systems (ISIS) Initiative**  
Unlimited distribution subject to the copyright.



This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2008 Carnegie Mellon University.

**NO WARRANTY**

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be directed to the [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (<http://www.sei.cmu.edu/publications/pubweb.html>).

---

## Table of Contents

<b>Abstract</b>	<b>vii</b>	
<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Basic SOA Concepts</b>	<b>2</b>
<b>3</b>	<b>Challenges of Migration to SOA Environments</b>	<b>3</b>
3.1	Legacy System Challenges	3
3.2	SOA Environment Challenges	4
<b>4</b>	<b>Service Migration and Reuse Technique (SMART)</b>	<b>6</b>
4.1	Four Elements of SMART	6
4.2	The SMART Process	7
4.2.1	Establish Context	8
4.2.2	Migration Feasibility Decision Point	8
4.2.3	Define Candidate Services	9
4.2.4	Describe Existing Capability	9
4.2.5	Describe Target SOA Environment	10
4.2.6	Analyze the Gap	10
4.2.7	Develop Strategy	11
<b>5</b>	<b>Application of SMART to a Mission Status System</b>	<b>13</b>
5.1	Establish Context	13
5.2	Migration Feasibility Decision Point	14
5.3	Define Candidate Services	14
5.4	Describe Existing Capability	14
5.5	Describe Target SOA Environment	15
5.6	Analyze the Gap	16
5.7	Develop Strategy	17
<b>6</b>	<b>Conclusions and Next Steps</b>	<b>20</b>
<b>Appendix A</b>	<b>The Service Migration Interview Guide (SMIG)</b>	<b>22</b>
A1.	Establish Context	22
A2.	Preparation for Next Steps	25
A3.	Define Candidate Services	25
A4.	Describe Existing Capabilities	26
A5.	Describe Target SOA Environment	28
<b>Appendix B - The SMART Tool</b>		<b>31</b>
B1.	SMART Tool Components	31
B1.1.	SMART Client	31
B1.2.	SMART Server	32
B2.	Tool Usage Scenario	33
<b>References</b>		<b>35</b>



---

## List of Figures

Figure 1:	High-Level Representation of a Service-Oriented System	2
Figure 2:	The SMART Process	7
Figure 3:	Notional Architecture for the Service-Oriented System Based on MSS	16
Figure 4:	Service Reference Architecture for MSS Services	18
Figure 5:	SMART Family	21
Figure 6:	Screenshot of the SMART Client	32
Figure 7:	Screenshot of the SMART Server	33



---

## List of Tables

Table 1:	SMART Activities and Artifacts	12
Table 2:	Options for Short-Term Feasibility Demonstration	17
Table 3:	Suggested Migration Iterations	17
Table 4:	Business and Technical Context	23
Table 5:	Stakeholders	23
Table 6:	Legacy System and Target SOA Environment	24
Table 7:	Candidate Service Identification	24
Table 8:	Define Candidate Services	25
Table 9:	Legacy System Characteristics	26
Table 10:	System Architecture	27
Table 11:	Code Characteristics	27
Table 12:	Target SOA Environment Characteristics	28
Table 13:	Support	30



---

## Abstract

Service-oriented architecture (SOA) has become an increasingly popular mechanism for achieving interoperability between systems. Because it has characteristics of loose coupling, published interfaces, and a standard communication model, SOA enables existing legacy systems to expose their functionality as services, presumably without making significant changes to the legacy systems. Migration of legacy systems to service-oriented environments has been achieved within a number of domains—including banking, electronic payment, and development tools—showing that the promise is beginning to be fulfilled.

While migration can have significant value, any specific migration requires a concrete analysis of the feasibility, risk, and cost involved. This technical note describes a new release of the Service Migration and Reuse Technique (SMART), which was initially developed in 2005. The Carnegie Mellon® Software Engineering Institute (SEI) SMART process helps organizations to make initial decisions about the feasibility of reusing legacy components as services within an SOA environment. SMART considers the specific interactions that will be required by the target SOA environment and any changes that must be made to the legacy components. To achieve this, SMART gathers information about legacy components, the target SOA environment, and candidate services to produce (1) a preliminary analysis of the viability of migrating legacy components to services, (2) an analysis of the migration strategies available, and (3) preliminary estimates of the costs and risks involved in the migration.



---

## 1 Introduction

Service-oriented architecture (SOA) has become an increasingly popular mechanism for achieving interoperability between systems. Because it has characteristics of loose coupling, published interfaces, and a standard communication model, SOA enables existing legacy systems to expose functionality as services, presumably without making significant changes to those systems. Migration of legacy components to services has been achieved in a number of domains—including banking, electronic payment, and development tools—showing that the promise is beginning to be fulfilled [Chung 2005, Polmann 2002, Radha 2004, Zhang 2004]. While migration can have significant value, any specific migration requires a concrete analysis of the feasibility, risk, and cost involved.

This report discusses the role of SOA and presents the Carnegie Mellon® Software Engineering Institute (SEI) Service Migration and Reuse Technique (SMART)—a technique to help organizations make initial decisions about the feasibility of reusing legacy components as services within an SOA environment. SMART was initially developed in 2005 [Lewis 2005, Lewis 2006]. The version of SMART outlined in this report represents a significant new release that has been revised based on experience with the process.

Section 2 of this report provides an overview of basic SOA concepts. Section 3 presents the challenges of migrating legacy components to an SOA environment. Section 4 presents the SMART process. Section 5 presents a case study of SMART applied to a real migration project. Finally, Section 6 provides conclusions and next steps.

---

<sup>®</sup> Carnegie Mellon is registered in the U. S. Patent and Trademark Office by Carnegie Mellon University.

## 2 Basic SOA Concepts

SOA is a way of designing systems composed of services that are invoked in a standard way. SOA is an architectural style—it is neither a system architecture nor a complete system. At a high level, a service-oriented system is composed of

- Services: reusable components that represent business or mission tasks, such as customer lookup, weather, sensor placement, account lookup, or credit card validation. Services can be globally distributed across organizations and reconfigured to support new tasks or missions. They are reusable because they can be used by a number of business processes or mission threads. They usually provide coarse-grained functionality, such as customer lookup, as opposed to finer-grained functionality such as customer address lookup.
- Service Consumers: These are clients for the functionality provided by the services. Some examples of service consumers are end-user applications, portals, internal or external systems, or even other services in the context of composite services. In a typical business setting, an order processing application may use services such as customer lookup, credit check, and item lookup that are derived from a number of sources inside and outside the enterprise.
- SOA Infrastructure: The infrastructure connects service consumers to services. It usually implements a loosely coupled, synchronous or asynchronous, message-based communication model, but other mechanisms are possible. The infrastructure often contains elements to support service discovery, security, and other operations. A common SOA infrastructure is an Enterprise Service Bus (ESB) to support web service environments. The Army System of Systems Common Operating Environment (SOSCOE) and Defense Information Systems Agency (DISA) Net-Centric Enterprise Services (NCES) are two examples of SOA infrastructures within the U. S. Department of Defense (DoD).

A notional representation of a service-oriented system is shown in Figure 1.

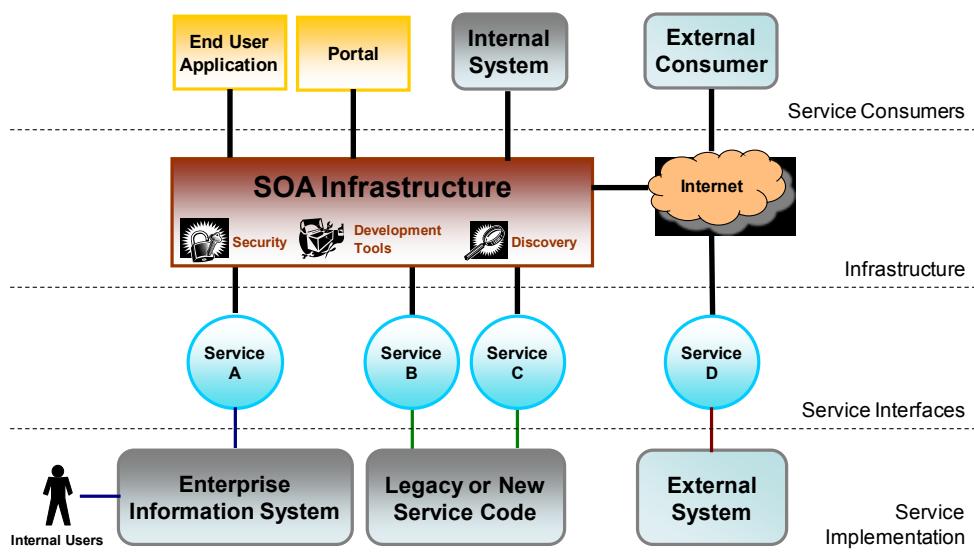


Figure 1: High-Level Representation of a Service-Oriented System

---

### 3 Challenges of Migration to SOA Environments

One of the most attractive promises of an SOA environment is that it enables reuse of legacy systems, thereby providing a significant return on the investment in these systems. However, migrating legacy systems is neither automatic nor easy. Traditional reuse challenges apply to SOA environments, but those challenges are heightened in both positive and negative ways by the granularity of what is exposed through reuse [Morisio 2002]. Reuse in the SOA context is typically most effective when the services correspond to coarse-grained business or mission functionality, such as order placement or flight path calculation where all underlying technical details are encapsulated by a standard service interface.

Service consumers and service providers experience the effect of legacy system migration in different ways. Service consumers benefit because functionality can be acquired, rather than developed, leading to potential savings. However, those acquired services have to be used “as-is.” If the available services do not match their needs, service consumers may see incompatibilities with their existing business processes. Service providers face the challenge of providing a service that applies to many service consumers, yet still adds value. The following sub-sections provide greater detail of the nature of these challenges.

#### 3.1 LEGACY SYSTEM CHALLENGES

Because of technical constraints, it may not always be possible to reuse functionality of legacy systems by exposing it as a service. Some of these technical constraints stem from the nature of the legacy system, and others are because of immature technology for a particular legacy environment. As a result, the cost of exposing parts of a legacy system as services could be higher than actually replacing the legacy system with a new service-oriented system. Some situations in which it would be less expensive to replace than reuse are as follows:

- If user interface code is tightly coupled with business or mission function code, there will be a large amount of rework to separate out what is purely functional, given that services should be user-interface agnostic.
- If the target SOA environment is Web Services,<sup>1</sup> XML and SOAP libraries may not available for all legacy platforms.
- The synchronous behavior of the legacy system may be in conflict with the asynchronous nature of SOA environments.
- A batch-oriented legacy system may be in conflict with the request-response nature of SOA environments where a user expects a close-to-immediate response.
- An organization might run into licensing issues with underlying commercial products where functionality is now exposed to a greater number of consumers, potentially outside the organization.

---

<sup>1</sup> The most common (but not *only*) form of SOA implementation is that of web services, in which (1) service interfaces are described using Web Services Description Language (WSDL), (2) payload is transmitted using Simple Object Access Protocol (SOAP) over Hypertext Transfer Protocol (HTTP), and, optionally, (3) Universal Description, Discovery and Integration (UDDI) is used as the directory service.

To make effective decisions, people managing legacy system migration to SOA environments need to identify relevant and non-relevant legacy components and choose which ones to investigate through “hands-on,” contextual analysis. In support of the decision-making process, those individuals need estimates of cost and risk, as well as confidence in those estimates, for each legacy component.

### 3.2 SOA ENVIRONMENT CHALLENGES

The complexity of the migration will largely depend on the characteristics of the SOA environment. Some examples of those characteristics are as follows:

- The user community for a service-oriented system can be known, as in the case of services exposed within a single organization. Or the user community can be unknown, as in the case of services exposed to the general public via the internet. It stands to reason that the larger and more unknown the community, the larger and more complex the migration challenges. A number of questions need to be addressed, including the following:
  - How will the services be used? What information is expected to be exchanged? In what format?
  - What is the right granularity for the service? How generic should it be?
  - Will the services scale to the size of the user community? How will performance be affected?
  - What security measures need to be taken, given the nature of the user community?
  - What operational procedures need to be in place?
  - Once services are deployed, what is the procedure for change management? How can changes be promulgated to potentially unknown users?
- There are many ways to implement service-oriented systems. On one end of the spectrum, there are basic implementations, typically based on widely available technologies and standards such as web services. On the other end of the spectrum, there are proprietary implementations where certain technologies are selected to satisfy specific provider or consumer requirements such as performance or security. These proprietary environments will require greater effort to understand the technologies involved, tool availability, and constraints placed on service consumers and providers.
- The rationale for the migration to services might be to eliminate redundant functionality and data through data services or a shared data model. If this is the case, a number of questions need to be addressed, such as
  - Will data be accessible only through new data services?
  - Will legacy systems that will continue to work stand alone need to be modified to access the new data services?
  - Does the shared data model contain all of the data needed by the legacy components?
  - Are the current and shared data models compatible?
- A stand-alone system with a current set of users can become a component of a system of systems by exposing services. If this is the case, the system now has two sets of users—internal users and service consumers. This creates potential for conflicting requirements, more complex change management procedures, and performance degradation.

These potential issues highlight the need for an upfront and hands-on analysis of technical feasibility and the resultant return on investment in order to avoid last-minute surprises. The issues involved go beyond adding a service interface to an existing system.

---

## 4 Service Migration and Reuse Technique (SMART)

SMART is an approach for making decisions on the migration of legacy components to services. It analyzes the viability of reusing legacy components as the basis for services by answering these questions:

- Does it make sense to migrate the legacy system to services?
- What services make sense to develop?
- What components can be mined to derive these services?
- What changes are needed to accomplish the migration?
- What migration strategies are most appropriate?
- What are the preliminary estimates of cost and risk?

### 4.1 FOUR ELEMENTS OF SMART

SMART consists of four elements:

1. The SMART Process is a systematic means to gather information about the legacy components, the candidate services, and the target SOA environment.
2. The Service Migration Interview Guide (SMIG) guides the discussions during the initial SMART process activities. It contains more than 60 categories of questions that gather information about the migration context, the legacy components, the candidate services, and the target SOA environment. The goal of using the SMIG is to assure broad and consistent coverage of the factors that influence the cost, effort, and risk in migration to services. Each question in the SMIG is associated with potential migration issues or aspects that are known to require extra cost or effort. A representative subset of the SMIG is included in Appendix A.
3. Using the SMIG as a framework, the SMART Tool automates data collection and relates answers to questions to potential risks to mitigation strategies. Then, answers and associated information yield a draft migration strategy and migration issues list. The tool also consolidates data from multiple engagements for trend analysis. The SMART Tool is described in Appendix B.
4. Artifact Templates for output products are created as part of the process. These templates, which are initially populated by the SMART Tool, include the following:
  - Stakeholder List: Contains the information about all stakeholders who will provide input into the process—sponsors, managers, system developers, system maintainers, system architects, representatives of service consumers, and IT staff
  - Characteristics List: Contains the list of characteristics that needs to be gathered about each component targeted for migration. It initially contains basic information such as name, function, size, language, operating platform, age and gets updated as migration issues are identified.
  - Migration Issues List: Contains the list of migration issues that are identified during the information-gathering activities

- Business Process-Service Mapping: Contains the mapping between main business processes and candidate services
- Service Table: Contains information about candidate services such as description, associated legacy components, inputs, and outputs
- Component Table: Contains information about legacy components targeted for migration as identified in the Characteristics List
- Notional Service-Oriented System Architecture: Presents a high-level view of the system architecture showing service consumers, infrastructure components, services, and legacy components, as well as their interaction
- Service-Component Alternatives: Presents the different options for satisfying candidate service requirements. Options are wrap, extract, create new, rewrite in a different language, add external service, acquire commercial product, or fashion any combination of the above.
- Migration Strategy: Contains the migration strategy for the targeted legacy components, as well as guidance for future migration efforts

## 4.2 THE SMART PROCESS

The SMART process has six activities and one major decision point, as illustrated in Figure 2. The activities are iterative: data gathered in one activity may provide questions that require revisiting an earlier activity for additional information. The following sub-sections outline each of these activities.

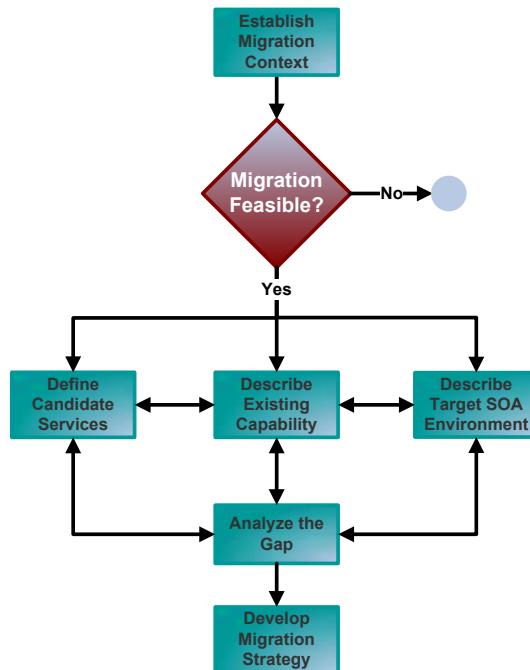


Figure 2: The SMART Process

#### 4.2.1 Establish Context

The *Establish Context* activity has the following tasks:

- Understand the business and technical context for migration. In this activity, information is gathered about the rationale, goals, and expectations for migration to an SOA environment, the technical and business drivers, programmatic constraints such as budget and schedule, and any previous related efforts or analyses.
- Identify stakeholders. Information is gathered to identify who (1) is driving and paying for the effort, (2) knows about the legacy system and the target SOA environment (and what they know), and (3) creates the demand or need for potential services.
- Understand the legacy system and target SOA environment at a high level. Basic information about the legacy system is gathered, such as main functionality, size, technologies, age, history, and users. Of interest about the target SOA environment at this point are status, technologies, main components, and history.
- Identify a set of candidate services for migration. The selection of candidate services is both a top-down and a bottom-up approach guided by business or mission goals and the functionality that exists in the legacy system, as indicated by the following steps:
  1. Identify business or mission goals
  2. Identify key business processes or mission threads that support these goals
  3. Identify common steps or tasks in these processes or threads
  4. Identify functionality from the legacy system to support these steps/tasks
  5. Negotiate to select a number of the steps as candidate services

During the *Establish Context* activity, the following artifacts are initially developed:

- stakeholder list
- migration issues list
- characteristics list
- business process-service mapping

Because SMART features an iterative process, these artifacts are updated through the rest of the activities as additional information is gathered, as Table 1 on page 12 shows.

#### 4.2.2 Migration Feasibility Decision Point

After the *Establish Context* activity, there is an explicit decision point to determine if the legacy system is a good candidate for migration to services (denoted in Figure 2 by the diamond shape labeled *Migration Feasible?*). If the legacy system is not a good candidate, stopping at this point will save time and money. A decision to stop is a positive outcome of the SMART analysis because it preserves valuable resources for other activities.

Potential determinations are

- There is enough migration potential to continue the analysis.
  - Migration goals are clear and shared among stakeholders.
  - There is a high-level understanding of the legacy system and the target SOA environment.

- Candidate services and potential service consumers have been identified.
  - A very preliminary mapping of services to legacy components has been done.
- The migration has potential but requires additional information to make an informed decision and continue with the SMART process. This additional needed information may include
  - greater articulation of business goals needed to in order to clearly understand what is expected from the migration
  - identification of potential service consumers in order to provide a clear justification of the need for the services
  - availability of key stakeholders to support the process: project sponsors, legacy system developers/maintainers, future service developers, and target SOA environment owners
  - identification of target SOA environment
- The migration is not feasible. Some indications that the migration is not feasible are
  - There are no identifiable consumers for the services to be migrated from the legacy system.
  - Functionality in the legacy system does not have potential for use by multiple consumers.
  - No functionality in the legacy system of a stateless nature.<sup>2</sup>
  - Adequate input for the candidate services would require the construction of very complex applications.
  - There appears to be incompatibility between the legacy system and the target SOA environment.

#### 4.2.3 Define Candidate Services

Provided that a decision is made that migration is feasible, the process continues with an activity to define the candidate services. The goal of this activity is to select a small number of services (usually 3 to 4) from the initial list of candidate services that were identified as part of the *Establish Context* activity. Good candidate services are ones that perform concrete functions, have clear inputs and outputs, and can be reused across a variety of potential applications. These candidate services are now specified more completely to include a definition of service inputs and outputs, and quality of service (QoS) requirements.

The Service Table artifact is created during this activity and updated in the other information gathering activities that occur in parallel—*Describe Existing Capability* and *Describe Target SOA Environment*. (See Table 1 on page 12.)

#### 4.2.4 Describe Existing Capability

The goal of this activity is to gather information about the legacy system components that contain the functionality meeting the needs of the services selected in the *Define Candidate Services* activity. Technical personnel are questioned about system aspects such as

- descriptive data about legacy components—name, function, size, language, operating platform, age

---

<sup>2</sup> In a request-response mode, a stateless nature means that no variables need to be maintained between requests. It does not mean that there cannot be a state change within the legacy system, such as a change in the information stored in a database.

- architecture views
- design paradigms
- system quality
- change history
- user satisfaction
- existing problems

Additional information needed about components will be determined by the migration issues that emerge during the process. For example, if the legacy system has dependencies on commercial products that potentially may experience problems in the target service-oriented environment, it is important to know if the specific components targeted for migration share those dependencies. An analysis of options for dealing with these dependencies is determined during the *Analyze the Gap* activity.

The Component Table artifact is created during this activity and updated in the other information gathering activities as needed. (See Table 1 on page 12.)

#### **4.2.5 Describe Target SOA Environment**

This activity gathers information about the target SOA environment for the selected services including

- major components of the SOA environment
- impact of specific technologies and standards used in the environment
- guidelines for service implementation
- state of target environment
- interaction patterns between services and the environment
- QoS expectations and execution environment for services

As we mentioned earlier, all SMART activities are iterative. Information gathered during this activity may also trigger additional information that needs to be gathered about components. For example, if the target SOA environment contains a component for information security management, it is important to identify whether any of the components targeted for migration will need to make use of this security component. The specific options for integration with the security components are determined during the *Analyze the Gap* activity.

A Notional Service-Oriented System Architecture artifact, similar to that in Figure 1 on page 2, is created during this activity to illustrate the components of the system—service consumers, infrastructure, services, legacy components—and how they interact with each other.

#### **4.2.6 Analyze the Gap**

This activity provides preliminary estimates of the effort, risk, and cost to convert the candidate legacy components into services, given the candidate service requirements and target SOA characteristics. The discussion of the changes that are necessary for each component is used as the input to calculate these preliminary estimates.

In some cases, additional analysis methods may be needed, such as evaluation of code quality using code analysis tools or architecture reconstruction. For example, if the dependencies between

components of the system are not well known and the technical personnel is not capable of providing details of the changes or the magnitude of the changes, an architectural reconstruction could provide a set of views to understand these dependencies [Kazman 2002, O'Brien 2002].

The Service-Component Alternatives artifact is created during this activity to illustrate the potential sources for functionality to satisfy service requirements. (See Table 1.)

#### **4.2.7 Develop Strategy**

The information gathered in the previous activities generates migration issues that need to be addressed by the migration strategy. This information also provides the basis for estimates of cost, effort, and risk of migration, which will place constraints on the migration strategy. This activity develops a migration strategy that may include

- feasibility, risk, and options for proceeding with the migration effort
- identification of a pilot project to migrate a simple service (or set of services) that has high visibility and low risk, especially if the organization is new to SOA. This allows the organization to become familiar with the technologies, gain organizational buy-in, and start defining processes for later service development.
- order in which to create additional services
- guidelines for identification and creation of services. This includes
  - any specific guidelines to address particular migration issues—design patterns, specific technologies, infrastructure usage
  - service reference architectures. If there are unknowns about the infrastructure, data sources, system interfaces, and any other element that the services will interact with or if there is reason to believe that these elements are unstable or in constant change, it is important to architect the service in such a way that they are isolated from these changes. An example of a service reference architecture is presented in the case study in Section 5.
  - options for the source of service code—legacy system, commercial products, or external services
  - mechanisms for providing service functionality—wrapping, rewriting, extraction, or new
- specific migration paths to follow. A migration strategy may present a set of options for migration. For example, an approach may be to wrap the existing legacy code initially and rewrite the components in a different language in the future.
- needs for additional information or training. Any gaps identified by the migration issues need to be addressed—through, for instance, technology evaluation, market research, training, or workshops

Table 1: SMART Activities and Artifacts

	Establish Migration Context	Define Candidate Services	Describe Existing Capability	Describe Target SOA Environment	Analyze the Gap	Develop Migration Strategy
Stakeholder List	Create			Update		
Characteristics List	Create			Update		
Migration Issues List	Create			Update		
Business Process-Service Mapping	Create			Update		
Service Table		Create		Update		
Component Table			Create		Update	
Notional SOA-Based System Architecture				Create		Update
Service-Component Alternatives					Create	Update
Migration Strategy						Create

---

## 5 Application of SMART to a Mission Status System

The following is a summary of the application of SMART to a DoD Mission Status System. Each sub-section corresponds to a step in the SMART process.

### 5.1 ESTABLISH CONTEXT

A DoD organization has been tasked with developing services that can be used by mission planning and execution applications. As a transition organization, its goal is twofold: (1) develop the services and (2) become knowledgeable about migrating legacy systems to services in order to assist other organizations in doing so. The organization is engaged in several migration efforts but has not used a systematic approach for making decisions.

The Mission Status System (MSS) targeted for migration compares a planned mission against a current state to determine if corrective actions should be taken. The system obtains plan data and situational awareness data from a Planning System (PS). MSS and PS run on the same machine, and there is tight coupling between the two systems. Both MSS and PS are in the final stages of development and have not been deployed. A long-term business goal is the full migration of MSS to services. The technical driver is to make the developed services available to all planning and execution systems.

A standard web services environment has been selected as the target SOA environment for this pilot. The future environment for the developed services will most likely be a DoD proprietary SOA infrastructure. However, by performing and executing this pilot, the organization will gain valuable insights on the migration process. Also, the overall process, as well as at least a significant part of the analysis, can be carried forward. The goal for the pilot is to demonstrate, within four months, the feasibility of one exposing MSS component as a service to be used by one mission planning and execution system. The long-term goal is to migrate the full system to services in two years. Funding has been allocated for the full effort.

Representatives from MSS and from a mission planning and execution system that is a potential service consumer identified the following set of candidate services:

- AvailablePlans: provides a list of available plans that are being reasoned about
- TrackedTasksPerPlan: provides a list of tasks that are being tracked for a certain plan
- TaskStatus: provides the status for a given task in a given plan
- SetTaskAlert: alerts when a given task in a given plan satisfies a certain condition

These services were selected because their functionality is generic enough that it can be used by other known mission planning and execution systems.

Migration issues identified at this point are as follows:

- The short-term and long-term goals for the migration are different. The implication of this difference is that the work to accomplish the short-term goal might have to be redone to accomplish the long-term goal.
- The system is currently a single-user system. When capabilities are migrated to services, it will have to support multiple users.

- The system currently monitors a single plan. When capabilities are turned into services, it will have to support monitoring of multiple plans.

## 5.2 MIGRATION FEASIBILITY DECISION POINT

Based on the data obtained at this stage, a decision was made to continue with the rest of the SMART analysis. This was based on the following factors:

- the availability of stakeholders from the service provider and a service consumer
- a good understanding of MSS
- the request-response nature of the identified services
- a reasonable initial mapping of services to components

## 5.3 DEFINE CANDIDATE SERVICES

The list of services identified in the previous step was considered reasonable for analysis. Inputs and outputs were next identified in detail for each of these services.

Migration issues identified during this activity are as follows:

- The SetTaskAlert service implementation will require that (1) the alert is set up to respond to certain conditions and (2) the service consumer has to be notified via an event that the condition has been met. By contrast, service-oriented systems have typically been of a request-response nature, in which a service consumer sends a request for a task to be performed and a service provider performs the task and returns a response. The communication protocol between consumer and provider handles the exchange, and there are typically no special requirements on the consumer or the infrastructure other than support for the protocol. The handling of events in service-oriented environments has been recently introduced in SOA 2.0 [Violino 2007]. The implementation of SetTaskAlert will require that either the service provider or the infrastructure store the address of the service consumer so that it knows whom to notify and that the service consumer be set up to receive alerts.
- It is unclear how the alert mechanism is going to be implemented. The SOA infrastructure needs to call back the service consumer. The service consumer might have to set up a web service, which means it could not be a thin client (i.e., accessing the service application via a simple web browser without having to install a web server). There might also be firewall issues.
- The complexity of alert conditions is high. In MSS, this is currently done through the user interface. The service consumer interface will have to replicate this complexity, or conditions will have to be simplified or limited.

## 5.4 DESCRIBE EXISTING CAPABILITY

The following characteristics of the MSS were provided:

- MSS is in a demonstration state, rather than a production environment. There have been several prototypes and experiments to demonstrate its capabilities.
- MSS is written in C++, C#, and Managed C++ in a Visual Studio 2005 development environment. It runs on a Windows XP platform. The size of the full system is approximately 13,000 lines of code. The amount of code considered for migration depends on the scope of

the migration effort, although most of the code is being targeted for migration in the future. Code documentation was rated between *fair* and *good* by its developers.

- Several architecture views were presented that were useful for understanding the system: high-level context diagram, component-and-connector view, module view, and runtime view.
- As indicated previously, MSS relies on PS for plan data and situational awareness data. PS provides an interface for data exchange using XML. This is an advantage for future integration with PS when it becomes a service. However, there is a chance that the data models may not match.

Migration issues identified in this phase are as follows:

- Documentation for most of the analyzed classes was determined to be *fair*. As a result, documentation could be an issue if the system's original developers do not perform the migration.
- There is currently heavy communication between MSS and PS. It is unclear how performance will be affected when this communication takes place using services (recall that the two systems currently reside on the same machine).
- The task alert functionality is not currently implemented in MSS, and there are unknowns about the specifics of the implementation.

## 5.5 DESCRIBE TARGET SOA ENVIRONMENT

As mentioned earlier, the target SOA environment for the migration is a standard web services one. It was decided to use an existing setup based on Microsoft IIS and ASP.NET.

As also mentioned earlier, the SetTaskAlert service has two parts. The first part sets the alert conditions and the second part sends the alert to the service consumer when those conditions are met. Sending an alert requires knowledge of the address of the SetTaskAlert service consumer. It was decided to use an existing publish-subscribe component that currently runs on another of the organization's servers. It is a simple component where users subscribe to one or more pre-defined events and are notified when one of those events occurs. This component requires subscribers to be set up as web servers. The notional high-level architecture for the service-oriented system that is in the scope of this migration effort is presented in Figure 3.

Additional migration issues identified during this activity are as follows:

- It is not known whether the publish-subscribe component will allow someone to subscribe on behalf of a third party. If "subscription by proxy" is not allowed, the service consumer will have to be aware of its dependency on the publish-subscribe component in order to receive alerts. The ideal situation would be for the SetTaskAlert service code to subscribe on behalf of the service consumer, so that the service consumer is not affected if the alert mechanism changes.
- The service consumer would have to be set up as a web server that is configured to accept incoming messages from the publish-subscribe component. This configuration is a security concern, potentially.

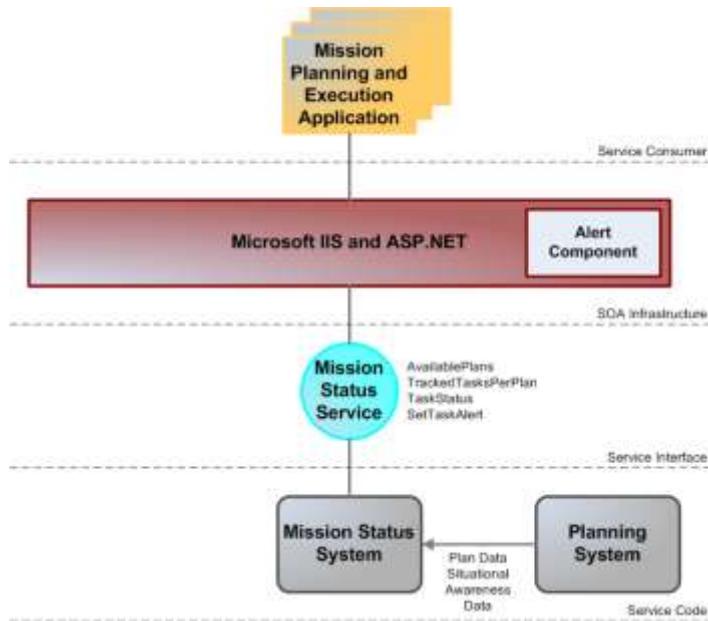


Figure 3: Notional Architecture for the Service-Oriented System Based on MSS

## 5.6 ANALYZE THE GAP

During this activity, the developers described the details of the changes that would have to be made to the code given the service requirements, the service inputs and outputs, and the characteristics and components of the target SOA environment. The developers were then asked to provide an estimate of the effort required to make these changes. No code analysis or architecture reconstruction was necessary because (1) the original developers were involved in the process, (2) their input was credible, and (3) the architecture documentation of and knowledge about the system were acceptable.

## 5.7 DEVELOP STRATEGY

Given the identified migration issues and preliminary estimates of cost and risk, the following migration strategy was developed.

1. **Define scope of initial migration for a short-term feasibility demonstration.** During the *Analyze the Gap* activity with developers of the legacy and the service consumer systems, the SMART team and the stakeholders discussed options for short-term feasibility experiments, as shown in Table 2.

*Table 2: Options for Short-Term Feasibility Demonstration*

Migration Option	Effort (person-weeks)
Implement SetTaskAlert service using a Query Language package developed for use in another system (Option 2)	24
Implement SetTaskAlert service using functionality in the legacy system (Option 1)	20
Do not implement the SetTaskAlert service	11
Do not implement the SetTaskAlert service and do not separate out from PS	7

The effort required has to be analyzed against the goals for the demonstration, and a decision about separating the service from PS has to be made. If the decision is not to separate the service from PS for this short-term feasibility demonstration, the group recommends it be done as part of a subsequent iteration in preparation for the long-term goal for MSS. The implementation of SetTaskAlert should meet the long-term goals for MSS and have the least impact on service consumers in terms of usability (from a service interface perspective) and performance.

2. **Define the scope of subsequent iterations.** A suggested set of iterations, according to input from the developers and other stakeholders as well as from recorded migration issues, is presented in Table 3. Subsequent iterations will depend on additional services to be created from MSS as well as progress made in the migration of PS to services.

*Table 3: Suggested Migration Iterations*

Iteration	Goal	Effort (person-weeks)
1	Implement AvailablePlans, TrackedTasksPerPlan, and Task Status	7
2	Separate MSS from PS	4
3	Implement SetTaskAlert (New code is needed for task alert.)	Option 1: 9 Option 2: 13
4	Add support for multiple users and multiple plans	TBD
5	Migrate to the DoD proprietary SOA environment	TBD

3. **Finalize service inputs and outputs.** The service inputs and outputs in the Service Table need to be concretely defined in WSDL documents, including the structure for conditions in SetTaskAlert that is still to be defined (for this or a future iteration, depending on scope selection).
4. **Gather information about the publish-subscribe component to be used as the mechanism for alert capability.** For the current or a future iteration, additional information about the publish-subscribe component to be used should be gathered to answer these questions:
  - Is it possible for the SetTaskAlert component to subscribe on behalf of the service consumer? If it is, the internet protocol (IP) address for the service consumer has to be passed as an input. If it is not, the service consumer has to be aware that it needs to subscribe to the publish-subscribe component.
  - What type of alert should SetTaskAlert or the service consumer subscribe to?
  - What are the requirements on the service consumer side to receive alerts?
5. **Create a reference architecture for the services.** A reference architecture to be followed by all services would provide a framework for service development, the reusability of common service operations, and, if done properly, the isolation of service code from changes due to the differences between short-term and long-term goals for MSS. An example of a service reference architecture is shown in Figure 4.

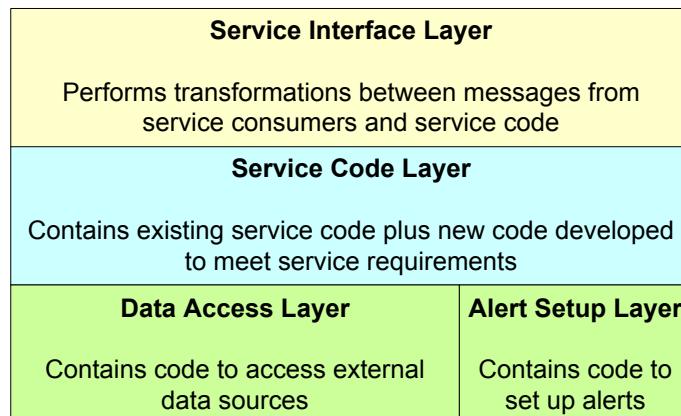


Figure 4: Service Reference Architecture for MSS Services

- Service Interface Layer: performs all transformations between messages from the service consumers and the MSS code, as well as input validation. This layer would isolate from changes in the evolution of the messages as the target SOA environment changes.
- Service Code Layer: contains all service functionality code, migrated or new.
- Data Access Layer: performs data access to all external sources. Initially, situational awareness data and plan data are external sources (even if currently done through a local application programming interface [API]). As PS migrates to a service, this layer would isolate existing code from incompatibilities between the current and future data structures.

- Alert Setup Layer: contains all code to setup the callback mechanism to the service consumer. This layer isolates code from changes if the selected publish-subscribe component is not a part of the future SOA infrastructure.

The implementation of the service reference architecture can be created as a project template in the selected development environment and used by all service development efforts.

6. **Adjust Estimates.** The estimates provided in the Component Table are based on a preliminary understanding of the inputs and outputs, as well as a high-level look at the code. After scope, inputs, outputs, and requirements are refined, the estimates will need to be adjusted.
7. **Create MSS services using the service reference architecture.** After defining the scope for the initial and subsequent iterations, migration and development should start as soon as possible to take advantage of MSS developer knowledge. In parallel with the migration and development, the service reference architecture should be implemented and refined.
8. **Document lessons learned.** Lessons learned in the process should be documented and published to support the goal of transition of SOA migration knowledge to other areas within the organization.

---

## 6 Conclusions and Next Steps

SOA offers significant potential for leveraging investments in legacy systems by providing a modern interface to existing capabilities, as well as exposing legacy functionality to a greater number of users. The SOA approach to systems development accomplishes this by promoting the assembly of applications from existing services, platform and language independence, reuse of services through loose coupling, and easy service upgrade due to separation of service interface from implementation.

There is a need for detailed analysis to determine the feasibility of exposing legacy functionality as services. One reason is that a service-oriented system consists of (1) services, (2) consumers that discover and use services, and (3) an SOA infrastructure that connects consumers to services. An end-to-end engineering approach for SOA requires addressing the unique challenges, risks, and technical issues of these three different development perspectives. The service provider that is designing reusable services, in particular, requires a different approach, skill set, and mindset than used in traditional development. In addition, there will be a bigger stakeholder community because services are typically reused at organization and sub-organization levels. Migration challenges may cause the cost of exposing legacy system functionality as services to be higher than actually replacing the system with a new service-oriented system. As a result, the detailed analysis has to include the identification of needs of the target SOA environment, a clear distinction between the needs that can be satisfied by the legacy system and those that cannot be satisfied, and a systematic analysis of changes that need to be made to fit into the target SOA environment.

Clearly, migration to SOA environments encompasses some complex engineering tasks. It requires an understanding of the role of SOA, potential pitfalls, and the unique challenges of migration within an SOA context. The type of data provided by the SMART approach enables an organization to make the initial decisions required for migration to a service-oriented environment. SMART analyzes the viability of reusing legacy components as the basis for services by answering these questions:

- Does it make sense to migrate the legacy system to services?
- What services does it make sense to develop?
- What components can be mined to derive these services?
- What changes need to be made to the components to accomplish the migration?
- What migration strategies are most appropriate?
- What are the preliminary estimates of cost and risk?

In just over three years, the SMART approach has been applied in four different organizations across six projects. As a result of these experiences, we have begun to identify variations on the SMART process to help organizations that are dealing with different sets of issues. The different variations of SMART are being built as part of a SMART Family, as shown in Figure 5. The members of the SMART Family follow the same process described in this report, but the emphasis is on certain activities in the process where the SMIG has been enhanced to go into more detail in specific areas.

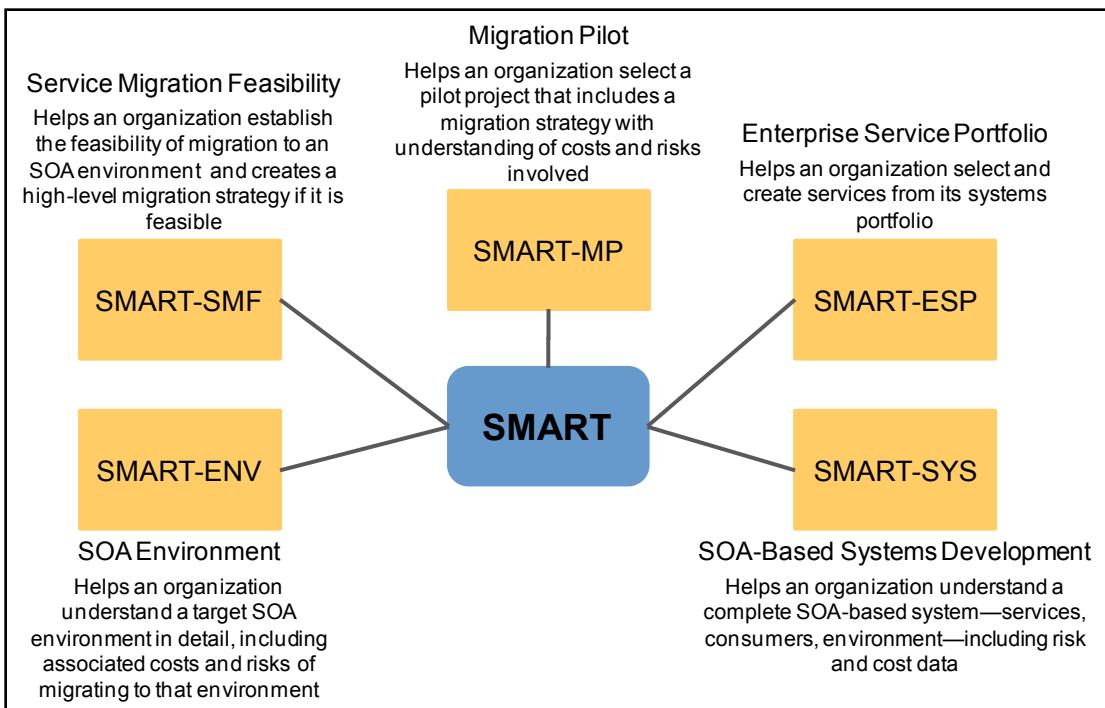


Figure 5: SMART Family

- SMART-MP (Migration Pilot) is the SMART process that was described in this report. The goal of SMART-MP is to identify a pilot project that will help shape a migration strategy for an organization, along with an understanding of cost and risk involved.
- SMART-SMF (Service Migration Feasibility) is tailored for organizations that are new to SOA and are probably not ready for a pilot project. The goal of SMART-SMF is to determine if it makes sense for an organization to adopt SOA, to understand its migration options, and to start putting together a migration strategy that may include the use of other members of the SMART Family.
- SMART-ENV (Environment) is aimed at organizations that have identified a target SOA environment (or have been mandated to use a particular SOA environment) but do not understand the implications of migrating to this environment. The goal for SMART-ENV is the characterization of the target SOA environment, including preliminary costs and risks of migrating to that environment.
- SMART-ESP (Enterprise Service Portfolio) enables organizations to scan across all of their legacy systems to identify potential services. The goal of SMART-ESP is the creation of an enterprise service portfolio and the mapping of these services to legacy systems.
- SMART-SYS (System) is targeted at organizations tasked with the development of a complete service-oriented system that potentially includes the identification and creation of services, the development or acquisition of an SOA infrastructure, and the development of service consumers. The goal in SMART-SYS is a superset of those of the previous SMART Family members.

The SMART Family will be outlined more completely in a future report.

---

## Appendix A The Service Migration Interview Guide (SMIG)

The SMIG is an instrument that focuses the discussions with stakeholders and developers in the first four activities of the SMART process:

- Establish Context
- Define Candidate Services
- Describe Existing Capabilities
- Describe Target SOA Environment

Answers to SMIG questions help determine the level of effort required to migrate legacy code into services. The use of this instrument assures broad coverage and consistent analysis of difficulty, risk, and cost issues.

### A1. ESTABLISH CONTEXT

The organization is asked to present

- Budget and schedule for the migration effort
- Business and technical drivers for the migration effort
- Characteristics of the organization that is performing the migration (if different)
- Characteristics of the organization that is sponsoring the migration effort
- Characteristics of the organization that owns the legacy system (if different)
- Characteristics of service consumers
- High-level architecture of the system
- High-level description of the system (functionality, history, users)
- High-level description of the target SOA environment
- List of candidate services (if available)
- Main business processes or mission threads that will be supported by these services (if available)
- Portions of the legacy system that contain the capabilities to support the candidate services (if available)

Discussion topics and questions that explore these areas are shown in the following tables.

*Table 4: Business and Technical Context*

Discussion Topic	Questions
Goal and Expectations of Migration	<ul style="list-style-type: none"> <li>• What are the business drivers for the migration effort?</li> <li>• Have any studies been conducted to verify these business drivers?</li> <li>• What are the technical drivers for the migration effort?</li> <li>• Are the technical drivers compatible with the business drivers?</li> <li>• What are the short-term goals of the migration effort?</li> <li>• What are the long-term goals of the migration effort?</li> <li>• Are the short-term and long-term goals compatible?</li> <li>• What are perceived advantages of migrating legacy components to services?</li> <li>• What are perceived disadvantages of migrating legacy components to services?</li> </ul>
Budget and Schedule	<ul style="list-style-type: none"> <li>• What is the timeframe for the migration?</li> <li>• Who is paying for the effort?</li> <li>• What is the budget for the migration?</li> </ul>
Other Migration Efforts	<ul style="list-style-type: none"> <li>• Have any other migration efforts been attempted?</li> <li>• What was the outcome?</li> <li>• Why did it fail or succeed?</li> <li>• What are the lessons learned?</li> </ul>

*Table 5: Stakeholders*

Discussion Topic	Questions
Legacy System End Users	<ul style="list-style-type: none"> <li>• Who are the end users of the legacy system?</li> <li>• Will legacy system end users be available during the migration process?</li> </ul>
Legacy System Owners	<ul style="list-style-type: none"> <li>• Who owns the legacy system?</li> <li>• If there is more than one owner, are these separate organizations?</li> <li>• Will legacy system owners be available during the migration process?</li> </ul>
Legacy System Developers and Maintainers	<ul style="list-style-type: none"> <li>• Who is the developer for the legacy system?</li> <li>• Are developers available to support the migration process?</li> <li>• Is the maintenance group separate from the development group?</li> <li>• If so, are maintainers available to support the migration process?</li> </ul>
Organization Performing the Migration	<ul style="list-style-type: none"> <li>• Are current developers or maintainers going to be performing the migration?</li> <li>• If not, what organization will perform the migration?</li> <li>• What is the process for bringing them up to speed on the legacy system?</li> <li>• Will this organization be available during the migration planning?</li> </ul>
Target SOA Environment Owners	<ul style="list-style-type: none"> <li>• Is the target SOA environment owned and maintained by a separate organization?</li> <li>• If so, will representatives be available to support the migration process?</li> </ul>

*Table 6: Legacy System and Target SOA Environment*

Discussion Topic	Questions
High-Level Understanding of Legacy System	<ul style="list-style-type: none"> <li>• What is the main functionality provided by the legacy system?</li> <li>• What is the history of the legacy system?</li> <li>• What is the high-level architecture of the system?</li> <li>• What portion of the system is envisioned for migration?</li> <li>• What is the current user interface to the legacy system?</li> <li>• How complex is the user interface?</li> <li>• What is the plan with respect to the legacy system(s)?</li> </ul>
High-Level Understanding of Target SOA Environment	<ul style="list-style-type: none"> <li>• What are the main components in the target SOA environment?</li> <li>• Is it a standard or proprietary environment?</li> <li>• Is this the organization's first attempt to deploy services in this environment?</li> </ul>

If the organization provides a list of candidate services, it is necessary to assess the process used to select them. An ideal process is to (1) identify business goals, (2) determine key business processes or mission threads that support these goals and can use functionality from the legacy system, and (3) find common steps/tasks in these processes or threads, and (4) select a number of the steps as candidate services. If the organization has not identified candidate services, the goal is to go through the process and identify some candidate services.

*Table 7: Candidate Service Identification*

Discussion Topic	Questions
Potential Services	<ul style="list-style-type: none"> <li>• Have potential services been identified?</li> <li>• If so, what was the process?</li> <li>• Is the list of services available?</li> </ul>
Potential Service Consumers	<ul style="list-style-type: none"> <li>• Who are the potential service consumers?</li> <li>• Have the potential service consumers provided both the business and the quality attribute requirements? In what form?</li> <li>• Are the identified service consumers internal or external to the organization?</li> </ul>
Business Goals and Processes Supported by Potential Services	<ul style="list-style-type: none"> <li>• What are the organization's main business goals to be supported by an SOA strategy?</li> <li>• What are the main business processes that support these goals?</li> <li>• What are common steps/tasks between these business processes?</li> </ul>
Initial Mapping Between Potential Services and Legacy Components	<ul style="list-style-type: none"> <li>• Has a mapping between services and components been done?</li> <li>• If so, is this mapping available?</li> <li>• What legacy components that provide the functionality are required by the services?</li> <li>• How different are the service requirements from the existing capabilities?</li> <li>• If there is a difference, how negotiable are the requirements?</li> </ul>

## A2. PREPARATION FOR NEXT STEPS

Once the migration is considered initially feasible, the next step is to gather additional detail on candidate services, the legacy system(s), and the target SOA environment. Stakeholders are asked to prepare the following:

- Detailed presentation(s) of requirements for services from real or potential service consumers
- Detailed presentation(s) of the legacy system, including all architectural views available
- Detailed presentation(s) of the target SOA environment, including technologies and any constraints the environment might place on service consumers and providers
- Legacy code for review on a laptop; LOC (lines of code) data for every legacy component

## A3. DEFINE CANDIDATE SERVICES

This activity selects a small number of services, usually three or four, from the initial list of candidate services. The goal is to fully specify inputs and outputs for these candidate services.

*Table 8: Define Candidate Services*

Discussion Topic	Questions
Service Consumers	<ul style="list-style-type: none"><li>• What specific applications or systems will be using these services?</li><li>• What is the expected service usage?</li><li>• What is the process for obtaining requirements from service consumers?</li><li>• What are specific quality attribute requirements, such as response time or security?</li><li>• Will formal/informal service level agreements need to be defined?</li></ul>
Refined List of Candidate Services	<ul style="list-style-type: none"><li>• What 3-4 services are the better match for the goals and expectations of the migration effort?</li><li>• What are the services with greater potential for use by service consumers?</li><li>• What are services with a better match to existing capabilities?</li><li>• What are the interfaces for these services in terms of inputs and outputs?</li></ul>
Mapping to Legacy Components	<ul style="list-style-type: none"><li>• For each service, what are the specific legacy components that contain the functionality required by the services?</li><li>• What new code will have to be written to fully satisfy service requirements?</li></ul>
Interface Negotiation	<ul style="list-style-type: none"><li>• What is the procedure for negotiating service interfaces with potential service consumers?</li><li>• How are conflicts to be solved?</li></ul>
Communities of Interest	<ul style="list-style-type: none"><li>• Is there a Community of Interest within the domain represented by the service capabilities?</li><li>• Are communities of interest internal or external to the organization?</li></ul>

#### A4. DESCRIBE EXISTING CAPABILITIES

In this activity, the SMART team and the client obtain descriptive data about the legacy system and its components. The goal is to capture basic characteristics of the legacy system, information about the architecture of the legacy system, and code characteristics that may affect the migration to determine whether it will be possible to use program-understanding tools for code analysis or architecture reconstruction, if necessary.

*Table 9: Legacy System Characteristics*

Discussion Topic	Questions
Functionality	<ul style="list-style-type: none"><li>• What is the main functionality provided by the system?</li></ul>
History	<ul style="list-style-type: none"><li>• What is the history of the system?</li><li>• How old is it?</li><li>• How many versions and releases have there been?</li></ul>
State	<ul style="list-style-type: none"><li>• Is the system a proof of concept, prototype, under development, in testing, or a fielded system?</li><li>• How stable is the system in general?</li></ul>
System Documentation	<ul style="list-style-type: none"><li>• What system documentation is available?</li><li>• How old is the documentation?</li><li>• What part of the system is not documented or has outdated documentation?</li></ul>
Size	<ul style="list-style-type: none"><li>• What is the size of the system?</li><li>• What is included in this number?</li><li>• What metrics are used in the size estimate for the system?</li></ul>
Platform	<ul style="list-style-type: none"><li>• What is the execution platform?</li><li>• Is it a distributed system? If so, have all system elements been included in the migration analysis?</li></ul>
Development Environment	<ul style="list-style-type: none"><li>• What is the development environment?</li></ul>
Interfaces with Other Systems	<ul style="list-style-type: none"><li>• Does the system have interfaces to other systems?</li><li>• Are these interfaces part of the code targeted for migration?</li><li>• Are interfacing systems aware of the migration effort?</li></ul>
System Users	<ul style="list-style-type: none"><li>• Is it a single-user or multi-user system?</li><li>• What are the potential locking, persistence, or transaction problems if accessed by multiple users when migrated to services?</li></ul>

Table 10: System Architecture

Discussion Topic	Questions
Architecture Documentation	<ul style="list-style-type: none"> <li>• What architecture views are available?</li> <li>• How old is the architecture documentation?</li> </ul>
Commercial Components	<ul style="list-style-type: none"> <li>• Are there dependencies on commercial components?</li> <li>• Is there support available for all commercial components?</li> <li>• How will the commercial components adapt to the services environment?</li> </ul>
Module View	<ul style="list-style-type: none"> <li>• What are the major modules of the system?</li> <li>• What are the dependencies between modules?</li> <li>• Does the code structure mimic this modular view?</li> </ul>
Deployment View	<ul style="list-style-type: none"> <li>• What is the deployment view of the system?</li> <li>• Are there dependencies on specific hardware or network topology?</li> </ul>
Runtime View	<ul style="list-style-type: none"> <li>• What is the runtime view of the system?</li> <li>• How is concurrency handled?</li> <li>• How are hard deadlines handled?</li> </ul>
Separation of Concerns	<ul style="list-style-type: none"> <li>• Is user interface code separate from the business logic code?</li> <li>• Is business logic code separate from middleware code?</li> <li>• Is data access logic separate from the rest of the code?</li> <li>• How tight is the coupling between these elements in the code?</li> <li>• What are other portions of the code where separation of concerns is a problem?</li> </ul>
Design Paradigms	<ul style="list-style-type: none"> <li>• Are there any design paradigms or patterns implemented in the system?</li> <li>• Are there any known violations to these paradigms due to tradeoffs with performance, for example?</li> </ul>
Quality Attributes	<ul style="list-style-type: none"> <li>• What are the key quality attributes built into the architecture of the system?</li> <li>• Are there any new quality attributes that are expected of the system when the services are in place?</li> </ul>

Table 11: Code Characteristics

Discussion Topic	Questions
Programming Language	<ul style="list-style-type: none"> <li>• What programming languages were used in the development of the system?</li> <li>• Do these languages have support for the technologies used in the target environment?</li> </ul>
Documentation	<ul style="list-style-type: none"> <li>• What code documentation is available?</li> <li>• Can the documentation be extracted using a tool such as Doxygen or JavaDoc?</li> </ul>
Coding Standards	<ul style="list-style-type: none"> <li>• What coding standards are followed?</li> <li>• Is the coding standards document available?</li> </ul>
Input Checking	<ul style="list-style-type: none"> <li>• Are there complete precondition and constraint checking on inputs?</li> </ul>
Code Organization	<ul style="list-style-type: none"> <li>• What is the code structure?</li> <li>• What is the mapping between the code structure and the module view of the system?</li> </ul>

## A5. DESCRIBE TARGET SOA ENVIRONMENT

This activity gathers information about the target SOA to support decisions about which services may be appropriate and how they will interact with the target SOA environment. The end goal is to produce a high-level notional architecture of the target service-oriented system.

### Target SOA Environment Characteristics

The goal of this section is to identify and gather sufficient detail about the target SOA environment to know how services will interact with the architecture and identify constraints or risks that may affect the migration effort. Potential conflicts between the legacy system components and the target architecture are also identified.

*Table 12: Target SOA Environment Characteristics*

Discussion Topic	Questions
Status	<ul style="list-style-type: none"><li>• What is the status of the target SOA environment?</li><li>• What builds are available? Are these the latest builds?</li><li>• What is the release schedule? Is it aligned with the migration schedule?</li></ul>
Communication with Target SOA Environment Organization	<ul style="list-style-type: none"><li>• If target SOA environment belongs to an external organization, what current communication and collaboration exists?</li></ul>
Infrastructure Components	<ul style="list-style-type: none"><li>• What are the major components of the SOA infrastructure?</li><li>• Which components are commercial and which will be developed internally?</li><li>• Is documentation available?</li><li>• How well specified is the infrastructure?</li></ul>
Infrastructure Services	<ul style="list-style-type: none"><li>• Does the target SOA environment provide infrastructure services (i.e., communication, discovery, security, data storage)?</li><li>• Is there redundancy between code internal to the services and the infrastructure services?</li><li>• Would it be feasible to replace internal calls with calls to the infrastructure services in the legacy code?</li></ul>
Communication Model	<ul style="list-style-type: none"><li>• What is the communication model(s) provided by the target SOA environment?</li><li>• Are there available libraries and tools in the legacy platform to support this communication model?</li></ul>
Standards and Mandates	<ul style="list-style-type: none"><li>• What are the standards or mandates that have to be followed?</li><li>• What is necessary to bring the legacy code in compliance with relevant standards and mandates?</li></ul>
SOA Environment Constraints	<ul style="list-style-type: none"><li>• What constraints does the target SOA environment impose on services?</li><li>• What are potential problems caused by these constraints (i.e., direct calls to the operating system)?</li><li>• Are there constraints on the use of commercial products?</li><li>• If there are problems, are there potential replacements?</li><li>• What are constraints on the granularity of the services due to, for example, deployment on limited memory devices?</li><li>• What are potential problems caused by this constraint (i.e., separating functionality into services)?</li></ul>

Table 12: Target SOA Environment Characteristics (contd.)

Discussion Topic	Questions
Architectural Mismatch	<ul style="list-style-type: none"> <li>• Does the legacy system have any behavior that would be incompatible with the target SOA environment, such as synchronous behavior, batch operation, or highly transactional activity?</li> <li>• What effort is required to eliminate or modify this behavior?</li> </ul>
Data Models	<ul style="list-style-type: none"> <li>• Does the target SOA environment impose a shared data model or a data infrastructure service?</li> <li>• What is the required effort to translate the legacy data model to the imposed data model?</li> <li>• What is the negotiation process for incompatibilities?</li> </ul>
Interfaces to Other Systems or Services	<ul style="list-style-type: none"> <li>• Does the target SOA environment provide interfaces to other systems or services?</li> <li>• Is there functionality in the legacy system that could be replaced by functionality in these systems or services?</li> <li>• What is necessary to use these systems or services or to prepare the system for future use of these systems or services?</li> </ul>
Service Description and Discovery	<ul style="list-style-type: none"> <li>• What are the requirements for description of services (e.g., WSDL for web services)?</li> <li>• What are the requirements with respect to making service descriptions available to potential consumers (e.g., UDDI, NCES directory service, appropriate agreements, other registries)?</li> </ul>
Ontologies	<ul style="list-style-type: none"> <li>• Is there a requirement for an ontology to be used in the description of services (e.g., OWL Web Ontology Language or other ontologies within the DoD Metadata Repository and Clearinghouse)?</li> <li>• How mature is the ontology?</li> <li>• Is it widely used?</li> <li>• Is a proprietary ontology being developed in-house?</li> <li>• Are there other ontologies in the same domain?</li> </ul>
Quality of Service (QoS)	<ul style="list-style-type: none"> <li>• Is it required to document QoS expectations and promises, preconditions, and other necessary characteristics of a service not covered by the service description specification (e.g., performance, reliability, error rate)?</li> <li>• What is the maturity of the specifications in this area?</li> </ul>
Quality of Protection (QoP) (Security)	<ul style="list-style-type: none"> <li>• Is it required to document QoP expectations and promises, preconditions, and other necessary characteristics of a service not covered by the service description specification (i.e., authentication mechanism, security classification, certification and accreditation status)?</li> <li>• What is the maturity of the specifications in this area?</li> </ul>
Service Execution Platform	<ul style="list-style-type: none"> <li>• Once developed, where will services execute?</li> <li>• Will they be hosted or will they be delivered to be deployed as needed?</li> </ul>
Service Management	<ul style="list-style-type: none"> <li>• What startup and initialization code is required for the services?</li> <li>• Is remote administration for monitoring and upgrade required?</li> </ul>

## Support

This goal is to collect information and generate awareness of the effort that is required once the services are deployed for use.

*Table 13: Support*

Discussion Topic	Questions
Testing	<ul style="list-style-type: none"><li>• Is there a requirement to provide test scripts and/or test cases for the services and make them publicly available?</li><li>• Are there mandates regarding testing and certification for services (i.e., Global Information Grid [GIG]–Network-Centric Warfare [NCW] requirements)?</li><li>• Is there a requirement to create test instances for services?</li></ul>
Service Consumer Setup and Installation	<ul style="list-style-type: none"><li>• Is there a requirement to develop setup and installation procedures for potential service consumers?</li><li>• Will consumers require configuration files or other mechanisms for setup?</li></ul>
Problem Reporting and Feedback	<ul style="list-style-type: none"><li>• Is there a requirement to establish problem reporting and feedback mechanisms for service consumers?</li></ul>
Updates and Upgrades	<ul style="list-style-type: none"><li>• How will service consumers be informed of potential changes in service interfaces and down time due to upgrades or problems?</li></ul>
User Communities	<ul style="list-style-type: none"><li>• Is there a user community for service consumers (e.g., demonstrations, tutorials, support for organizations attempting to use the services)?</li></ul>

---

## Appendix B - The SMART Tool

### The SMART Tool

- automates the SMIG data collection instrument
- relates questions to answers to potential risks and migration issues, producing a draft migration strategy and migration issues list—this data model is the “heart” of the tool
- consolidates data from multiple engagements for future analysis

### **B1. SMART TOOL COMPONENTS**

The Tool was developed in Java using the Eclipse Integrated Development Environment (IDE). Multiple free and open source tools and libraries were used for encryption, reporting, advanced user interface capabilities, communication, and data management. The Tool has two major components—the SMART Client and the SMART Server.

#### **B1.1. SMART Client**

The SMART Client is a Java application built using the Eclipse Rich Client Platform (RCP). This application is intended to run on the laptop computer in offline mode during the engagement. The SMART Client guides the facilitator through the engagement by

- presenting the SMIG to the facilitator for reference during the engagement. For each question, there are potential answers, space for comments, and a pre-determined next question to guide the discussions.
- providing means to tag questions to indicate elements of importance during the engagement, such as the need to revisit any question, potential characteristics for the Component Table or Service Table, major areas of risk, and any other custom tags defined for the engagement.
- calculating the status of the engagement constantly, based on the number of questions that have been answered in each category
- identifying and showing risks, as questions and answers. A list of overall risks is shown in the bottom portion of the screen for reference.
- consolidating data on the SMART Server, when there are multiple facilitators in an engagement. The consolidated data can then be downloaded for a group view.

The SMART Client also has reporting capabilities, as will be seen in Section B2.

A screenshot from the client is shown in Figure 6. A MySQL database is used to store engagement data, which is encrypted to assure confidentiality. A password is set during installation of the Client, so that it can only be executed by authorized users. The application is the only point of access to the encrypted data.

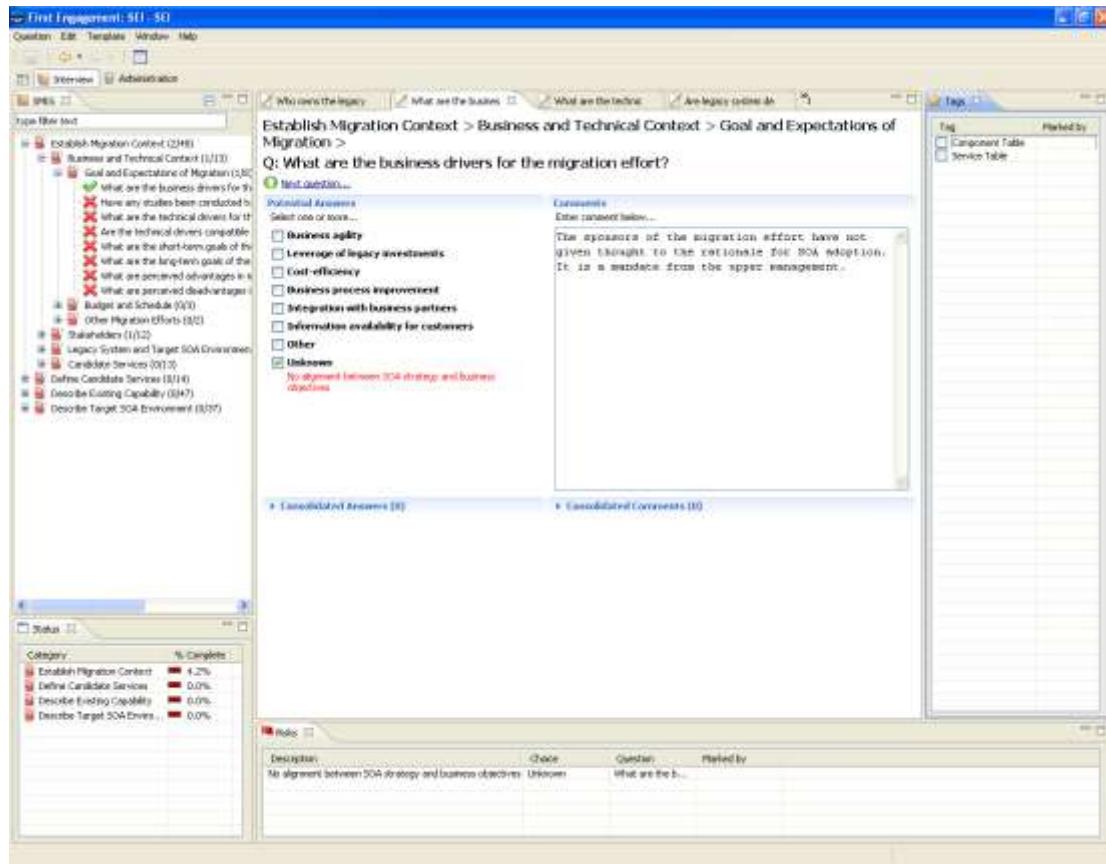


Figure 6: Screenshot of the SMART Client

### B1.2. SMART Server

The SMART Server is a web application that runs on a central server of an organization that performs SMART engagements. The SMART Server contains the following functionality:

- SMIG Maintenance: Allows the maintenance of SMIG elements: categories, questions, potential answers, risks, and mitigation strategies.
- Engagement Setup: Engagements are set up on the server and then downloaded by SMART Clients.
- User Management: This capability is for user and role management and assignment. Possible roles are facilitator, analyst, and administrator.
- Export and Import SMIG: A SMIG can be exported by an instance of a SMART Server and imported by another instance of the Server. This functionality is useful to disseminate SMIG updates.
- Reports: Reports are available for a version of the SMIG, a summary of an engagement, the final report for an engagement in Microsoft Word format, questions per tag per engagement, and on multiple engagements for analysis purposes.

A screenshot from the server user interface is shown in Figure 7. A MySQL database is used to store SMIG history and data from multiple engagements. Data is not encrypted on the server because the assumption is that the server resides inside the organization's firewall.



Figure 7: Screenshot of the SMART Server

## B2. TOOL USAGE SCENARIO

Assuming that both SMART Server and SMART Client are correctly installed, the following represents a typical usage scenario for a SMART engagement.

1. An engagement is set up by the SMART Administrator on the SMART Server. A version of the SMIG is associated with the engagement.
2. All SMART facilitators, using the SMART Client, connect to the SMART Server and download the engagement data and corresponding SMIG.
3. The SMART facilitators conduct the engagement using the SMART Client.
4. At the end of the first day, interview data is uploaded to the SMART Server by each facilitator using the SMART Client.
5. Before the start of the next day, consolidated interview data is downloaded by each of the SMART facilitators using the SMART Client.
6. On the SMART Client, one of the SMART facilitators produces a summary report of the engagement, indicating areas of disagreement between facilitators and tags these questions for later discussion with the team.
7. The SMART facilitators continue the engagement using the SMART Client, with access to comments and answers from other facilitators that provide greater insight.
8. At the end of the second day, interview data is uploaded again to the SMART Server by each SMART Client. (Steps 3 through 8 are repeated for each subsequent day of the engagement.)
9. On return to the SMART organization after the stakeholder interviews have been completed, the SMART analyst (an SEI team member or someone trained by the SEI team) exports data

produces the final report for the engagement for delivery to the client organization. This report is generated in Microsoft Word format for easy editing.

---

## References

URLs are valid as of the publication date of this document.

### [Chung 2005]

Chung, S., Young, P., & Nelson, J. “Service-Oriented Software Reengineering: Bertie3 as Web Services.” *Proceedings of the 2005 IEEE International Conference on Web Services (ICWS'05)*. Orlando, FL (USA), July 11–15, 2005. IEEE Computer Society, 2005. Digital Object Identifier 10.1109/ICWS.2005.109.

### [Kazman 2002]

Kazman, R., O'Brien, L., & Verhoef, C. *Architecture Reconstruction Guidelines, 2nd Edition* (CMU/SEI-2002-TR-034, ADA 421612). Software Engineering Institute, Carnegie Mellon University (2002). <http://www.sei.cmu.edu/publications/documents/02.reports/02tr034.html>

### [Lewis 2005]

Lewis, G., Morris, E., O'Brien, L., Smith, D., & Wrage, L. *SMART: The Service-Oriented Migration and Reuse Technique* (CMU/SEI-2005-TN-029, ADA441900). Software Engineering Institute, Carnegie Mellon University (2005). <http://www.sei.cmu.edu/publications/documents/05.reports/05tn029.html>

### [Lewis 2006]

Lewis, G., Morris, E., & Smith, D. “Analyzing the Reuse Potential of Migrating Legacy Components to a Service-Oriented Architecture.” *Proceedings of the 10<sup>th</sup> European Conference on Software Maintenance and Reengineering (CSMR 2006)*. Bari, Italy, March, 22–24, 2006. IEEE Computer Society, 2006. <http://ieeexplore.ieee.org/iel5/10671/33675/01602354.pdf?tp=&arnumber=1602354&isnumber=33675>

### [Morisio 2002]

Morisio, M., Ezran, M., & Tully, C. “Success and Failure Factors in Software Reuse.” *IEEE Transactions on Software Engineering* 28, 4 (April 2002): 340–357.

### [O'Brien 2002]

O'Brien, L., Stoermer, C., & Verhoef, C. *Software Architecture Reconstruction: Practice Needs and Current Approaches* (CMU/SEI-2002-TR-024, ADA407795). Software Engineering Institute, Carnegie Mellon University (2002). <http://www.sei.cmu.edu/publications/documents/02.reports/02tr024.html>

### [Polmann 2002]

Polmann, M. & Schonefeld, M. “An Evolutionary Integration Approach using Dynamic CORBA in a Typical Banking Environment.” Presented at the *Case Studies Workshop (CSW) of the Sixth European Conference on Software Maintenance and Reengineering (CSMR 2002)*. Budapest, Hungary, March 11–13, 2002. [http://www.omg.org/corba/industries/bankfin/gad\\_csmr\\_budapest\\_2002.pdf](http://www.omg.org/corba/industries/bankfin/gad_csmr_budapest_2002.pdf)

**[Radha 2004]**

Radha, V., Gulati, V., & Thapar, R. “Evolution of Web Services Approach in SFMS – A Case Study,” 640–647. *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*. San Diego, CA (USA), July 6–9, 2004. IEEE Computer Society, 2004.

**[Violino 2007]**

Violino, B. “How To Plan for SOA 2.0.” *Baseline* (March 8, 2007).  
<http://www.baselinemag.com/article2/0,1540,2102088,00.asp>

**[Zhang 2004]**

Zhang, J., Chung, J., & Chang, C. “Migration to Web Services Oriented Architecture—A Case Study,” 1624–1628. *Proceedings of the 2004 ACM Symposium of Applied Computing*. Nicosia, Cyprus, March 14 –17, 2004. ACM Press, 2004.

<b>REPORT DOCUMENTATION PAGE</b>		<i>Form Approved OMB No. 0704-0188</i>	
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</p>			
1. AGENCY USE ONLY  (Leave Blank)	2. REPORT DATE  June 2008	3. REPORT TYPE AND DATES COVERED  Final	
4. TITLE AND SUBTITLE  SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment		5. FUNDING NUMBERS  FA8721-05-C-0003	
6. AUTHOR(S)  Grace A. Lewis, Edwin J. Morris, Dennis B. Smith, Soumya Simanta			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER  CMU/SEI-2008-TN-008	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES			
12A DISTRIBUTION/AVAILABILITY STATEMENT  Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE	
13. ABSTRACT (MAXIMUM 200 WORDS)  Service-oriented architecture (SOA) has become an increasingly popular mechanism for achieving interoperability between systems. Because it has characteristics of loose coupling, published interfaces, and a standard communication model, SOA enables existing legacy systems to expose their functionality as services, presumably without making significant changes to the legacy systems. Migration of legacy systems to service-oriented environments has been achieved within a number of domains—including banking, electronic payment, and development tools—showing that the promise is beginning to be fulfilled. While migration can have significant value, any specific migration requires a concrete analysis of the feasibility, risk, and cost involved. This technical note describes a new release of the Service Migration and Reuse Technique (SMART), which was initially developed in 2005. The Carnegie Mellon® Software Engineering Institute (SEI) SMART process helps organizations to make initial decisions about the feasibility of reusing legacy components as services within an SOA environment. SMART considers the specific interactions that will be required by the target SOA environment and any changes that must be made to the legacy components. To achieve this, SMART gathers information about legacy components, the target SOA environment, and candidate services to produce (1) a preliminary analysis of the viability of migrating legacy components to services, (2) an analysis of the migration strategies available, and (3) preliminary estimates of the costs and risks involved in the migration.			
14. SUBJECT TERMS  Migration, legacy systems, service-oriented architecture, SOA, SMART		15. NUMBER OF PAGES  46	
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT  Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE  Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT  Unclassified	20. LIMITATION OF ABSTRACT  UL